Logistic Regression

Hoàng Vũ Minh

Logistic Regression là một kỹ thuật học có giám sát, tuy có regression ở tên nhưng được sử dụng chủ yếu cho các bài toán phân loại nhị phân, trong đó mục tiêu là dự đoán xác suất một điểm dữ liệu thuộc về một lớp cụ thể (thường là lớp positive, được ký hiệu là 1) hoặc lớp còn lại (thường là lớp negative, được ký hiệu là 0).

1. Ý tưởng cơ bản

Như đã trình bày ở trên, để phân loại các điểm dữ liệu về một trong hai lớp 0 1 cụ thể, ta cần biết giá trị xác suất thay vì giá trị. Vì vậy, cần sử dụng một activation function - hàm kích hoạt, trong Deep Learning, nó có tác dụng convert đầu vào cho các layer tiếp theo, còn ở đây, nó có tác dụng chuyển đầu ra thành một số thuộc khoảng (0,1) đại diện cho một giá trị xác suất.

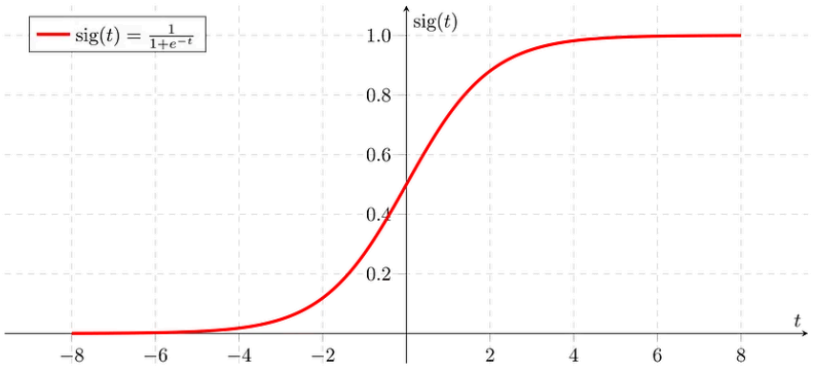
Diễn giải bằng hình học: Giả sử ta có dữ liệu như này, với Age Gender, Smoker status là các biến độc lập và DIsease (có bênh hay không) là biến phụ thuộc chỉ có 2 giá trị 0 hoặc 1.

Age	Gender	Smoker status	Disease
22	female	Non-smoker	1
25	female	Smoker	1
18	male	Smoker	0
45	male	Non-smoker	0
12	female	Smoker	0
43	male	Smoker	1
23	male	Smoker	0
33	male	Smoker	1

Khi plot ra, cho dù các giá trị của các biến độc lập như thế nào, kết quả của ta cần chỉ có 0 hoặc 1.

Như vậy, ta sẽ không thể sử dụng một đường thẳng hồi quy tuyến tính đi ngang qua làm đường dự đoán được vì nó không chính xác. Và ta cũng chỉ cần dự đoán giá trị trong khoảng 0 1, không cần một đường kéo dài vô tận, điều này thậm chí làm cho giá trị dự đoán đôi khi nằm ngoài khoảng. Vì vậy, ta cần phải sử dụng một hàm kích hoạt có tác dụng chuyển các giá trị trở thành giá trị trong khoảng 0 1 để phù hợp cho xác suất, hàm kích hoạt đó là sigmoid.

2. *Hàm sigmoid:*



Ta có z chính là phương trình hồi quy tuyến tính:

$$\sigma(u) = \frac{1}{1+e^{-z}} \quad \text{where} \quad z \in (-\infty, +\infty) \quad \text{and} \quad \sigma(u) \in [0,1]$$

$$\forall z_1, z_2 \in [a,b] \quad \text{and} \quad z_1 \leq z_2 \quad \rightarrow \quad \sigma(z_1) \leq \sigma(z_2)$$

Lý do sử dụng hàm sigmoid:

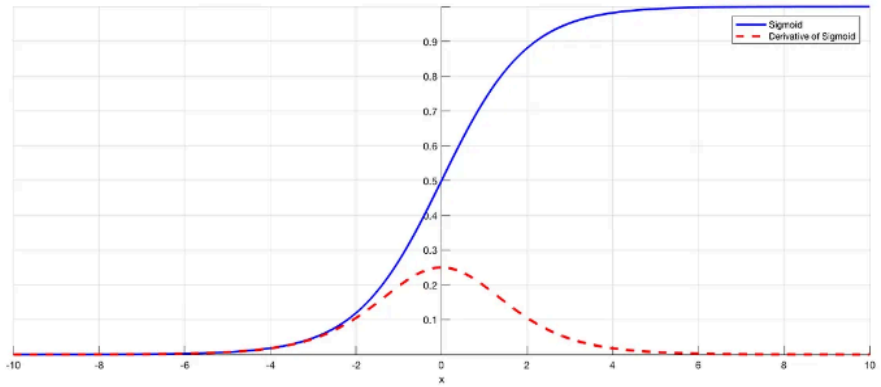
- ❖ Dễ đạo hàm, là hàm liên tục

- ❖ Bị chặn trong khoảng [0, 1], do đó phù hợp với các giá trị xác suất
- ❖ Chứa hàm số mũ của z:
  - ➢ - z ở dưới mẫu nằm trong hàm số mũ của e, tức là độ biến động của nó sẽ cao hơn so với bình thường, khi z khá lớn, -z sẽ nhỏ và exp(-z) sẽ trở nên nhỏ đi rất nhanh, khiến cho giá trị của hàm sigmoid gần về 1. Tương tự khi z giảm xuống khá nhỏ, giá trị của hàm sẽ nhanh chóng tiến dần về 0.
  - ➢ Tuy nhiên do hàm số mũ nằm ở dưới mẫu, nên biến động của nó cũng không quá lớn khi các input thay đổi không nhiều (nếu z rất lớn hoặc rất nhỏ), vì vậy nó có thể cho ra đầu ra khá "mượt".

Đạo hàm của hàm sigmoid:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Derivative of the Sigmoid Function



Từ đồ thị đạo hàm của hàm sigmoid, ta thấy nó cũng có một vài tính chất đặc biệt:

- Đạo hàm luôn dương -> sigmoid là một hàm đồng biến
- Đối xứng qua trục tung
- Có xu hướng bão hòa ở 2 đầu: Điều này vừa có lợi vừa có hại, tức là khi z rất lớn hoặc rất nhỏ, đạo hàm của hàm sigmoid sẽ gần như bằng 0. Có thể gây ra ảnh hưởng đến các quá trình cập nhật trọng số trong các mạng neural. Hiện tượng này gọi là Vanishing Gradient, sẽ được đề cập đến trong các phần lý thuyết về Deep Learning.

3. Cơ chế hoạt động:

Mô hình hồi quy logistic biến đổi đầu ra giá trị liên tục của hàm hồi quy tuyến tính thành đầu ra giá trị phân loại bằng cách sử dụng hàm sigmoid, hàm này ánh xạ bất kỳ tập hợp biến độc lập có giá trị thực nào đầu vào thành giá trị từ 0 đến 1.

Một ngưỡng log = 0.5 sẽ được chọn, nếu lớn hơn giá trị đó, ta sẽ coi là 1 và nhỏ hơn sẽ là 0. Tuy nhiên, ranh rới log = 0.5 này chỉ xuất hiện khi z= 0, trên thực tế điều này thường không hay xảy ra. Khi z lớn, giá trị hàm sigmoid sẽ tiến gần về 1 và ngược lại khi z nhỏ, nó sẽ tiến dần về 0. Điều này được phân tích ở phần trên, khi trình bày công thức hàm sigmoid.

4. *Hàm mất mát:*

Hồi quy logistic sử dụng Log Loss hay còn gọi là cross entropy loss (thật ra nó là một trường hợp đặc biệt của cross-entropy.

$$: -\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i) \log(1 - z_i))$$

**cross entropy, thường được sử dụng để đo khoảng cách giữa hai phân phối (distributions)*

Về phương pháp cực tiểu hóa hàm loss, có thể sử dụng SGD.

Giải thích thêm về hàm loss: Log-loss là một chỉ số cho thấy mức độ gần đúng của xác suất dự đoán so với giá trị thực tế/tương ứng (0 hoặc 1 trong trường hợp phân loại nhị phân), bằng cách phạt những dự đoán không chính xác với các giá trị cao hơn. Log-loss càng thấp thì hiệu suất của mô hình càng tốt, càng dự đoán gần với nhãn thực tế. Để cụ thể hóa, dựa vào công thức ta có thể phân tích:

$$: -\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i) \log(1 - z_i))$$

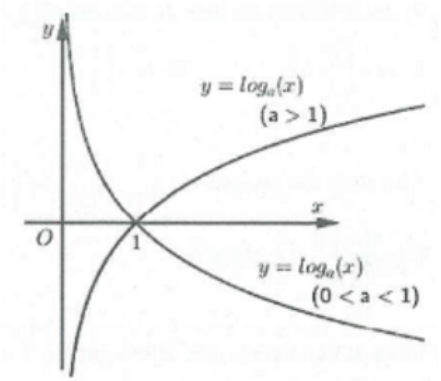
Ta có:

- Khi yi = 1 -> vế sau = 0, để tính giá trị loss ta chỉ care đến yi * log zi
    - Nếu zi cũng gần 1 (dự đoán gần như chính xác), log zi → 0 nhưng vẫn âm (nhìn đồ thị bên dưới) → yi log zi âm gần 0, gtri loss thêm dấu - sẽ thành dương gần 0, rất nhỏ, phạt ít.

- khi yi = 0 → vế đầu bằng 0, gtri loss = -log(1-z)
    - Nếu zi cũng gần 0, tức là 1-z gần 1, log gần 1 thì sẽ tiến đến 0, tương tự như trên, giá trị nó sẽ là dương gần 0, rất nhỏ.
- Còn nếu zi (xác suất dự đoán) mà gần 0.5, tức là không chắc chắn, điểm phạt sẽ tăng lên đáng kể.

Có thể hiểu rõ hơn dựa vào đồ thị của hàm log dưới đây:

- e≈2.718281828459045…
- Nên đồ thị ta nhìn vào sẽ là đường đi từ dưới lên trên. Nếu chỉ xét trong khoảng 0-1 (xác suất) thì: giá trị log luôn âm, tăng dần, bằng 0 tại điểm z = 1, khi z gần 0 thì giá trị log z rất nhỏ.

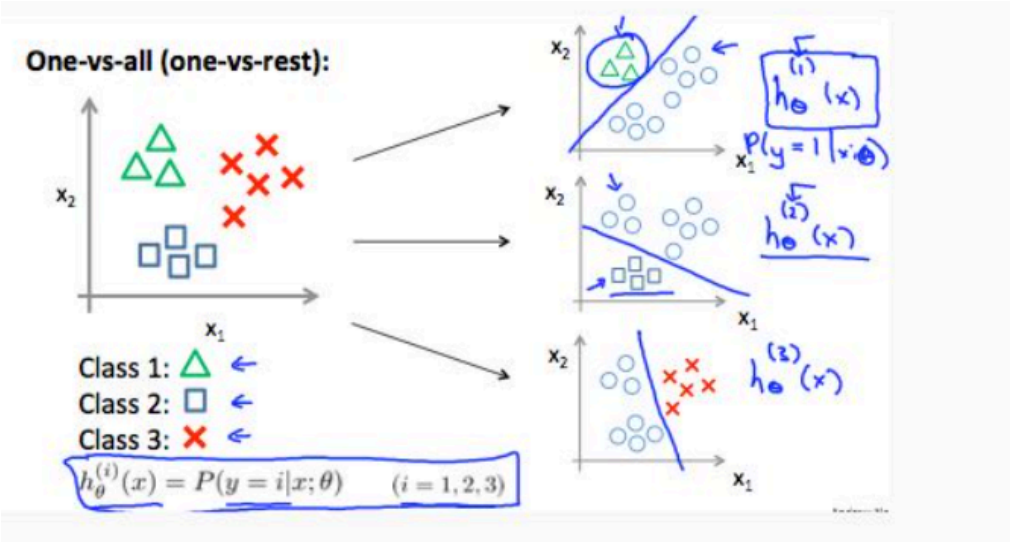


-

5. *Mở rộng: One vs Rest*

One vs Rest (từ đây trở đi em sẽ gọi tắt là OVR) là một kỹ thuật được sử dụng nhằm giúp các Binary Classifier (những mô hình phân loại nhị phân) có thể phân loại được nhiều nhãn. Ý tưởng cơ bản rất đơn giản, cho dù có bao nhiêu nhãn, ta cũng chỉ nó là 2 nhãn với nhãn 1 (tạm ký hiệu là O) với tất cả các nhãn còn lại.

Cụ thể, nếu có *C* classes thì ta sẽ xây dựng *C* classifiers, mỗi classifier tương ứng với một class. Classifier thứ nhất giúp phân biệt ***class 1*** và ***not class 1***, cứ tương tự như thế cho đến cuối cùng, ta sẽ thu được kết quả bằng cách xác định class mà xác suất của điểm đó rơi vào cao nhất.


One-vs-all (one-vs-rest):
x_2
x_1
Class 1:
Class 2:
Class 3:
$h_\theta^{(i)}(x) = P(y = i|x;\theta) \quad (i = 1, 2, 3)$
$h_\theta^{(1)}(x)$
$P(y = 1|x;\theta)$
$h_\theta^{(2)}(x)$
$h_\theta^{(3)}(x)$